XAML-merkintäkielen esittely

Ammattikorkeakoulututkinnon opinnäytetyö

Visamäki Hämeenlinna, Tietojenkäsittelyn koulutus

Kevät, 2017

Lauri Järvinen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

TIIVISTELMÄ

Tietojenkäsittelyn koulutusohjelma
Visamäki, Hämeenlinna

Tekijä Lauri Järvinen **Vuosi** 2017

Työn nimi XAML-merkintäkielen esittely

Työn ohjaaja/t Tommi Lahti

TIIVISTELMÄ

Opinnäytetyön aiheena oli toteuttaa XAMLin perusasioita käsittelevä esimerkkisovellus Androidille. Opinnäytetyö on tarkoitettu oppimateriaaliksi Mobile Programming -moduuliin tietojenkäsittelyn koulutusohjelmassa. Tavoitteena oli luoda toimiva sovellus, josta olisi helppo ymmärtää ja oppia XAMLin käyttötarkoitus ja peruselementit.

Toimeksiantajana tässä projektissa oli HAMKin opettaja Turo Nylund, joka työskentelee tietojenkäsittelyn koulutuksen Mobile Programming -moduulissa.

Opinnäytetyössä käsitellään XAMLia ja siihen liittyviä tekniikoita. Siinä esitellään myös mobiiliohjelmoinnin ja käyttöliittymien kehittämisen menetelmiä ja työkaluja.

Opinnäytetyön tuloksena on toimiva Android-sovellus, josta käyvät ilmi XAMLin perusasiat ja jonka avulla on helppo aloittaa käyttöliittymäkoodauksen opiskelu. Oppimateriaalista saa kattavan käsityksen XAMLista, mobiiliohjelmoinnista, tarjolla olevista työkaluista, käyttöliittymistä, käyttöliittymien suunnittelusta ja kehittämisestä.

Avainsanat XAML, C#, Xamarin, käyttöliittymä, mobiiliohjelmointi

Sivut 25 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Visamäki, Hämeenlinna

Author Lauri Järvinen **Year** 2017

Subject Introduction to XAML

Supervisors Tommi Lahti

ABSTRACT

The purpose of this thesis was to produce a simple application for Android, that introduces basic things about XAML. This thesis is made for Mobile Programming module, Degree Programme in Business Information Technology. The goal of this project was to create a working and simple application that helps learning XAML in the beginning.

The client of this project was Turo Nylund, teacher in HAMK. He works in the Mobile Programming module, Degree Programme in Business Information Technology.

In this thesis is presented XAML and its techniques. Also, this thesis presents some methods and tools you can use with mobile applications and user interfaces.

The result of this project was working Android-application, that introduces XAML and it truly helps when starting to learn user interface coding. Learning material gives wide information about XAML, mobile programming, tools, user interfaces, developing and planning user interfaces.

Keywords XAML, C#, Xamarin, User Interface, Mobile Programming

Pages 25 pages

SISÄLLYS

TERMIT JA SANASTO

SDK	Software Development Kit on ohjelmointipaketti, jonka avulla kehitetään sovelluksia erilaisille alustoille.
JDK	Java Development Kit on Javan ohjelmointipaketti, jota käytetään Java-pohjaisten sovellusten virheentarkistuksessa ja ajamisessa.
NDK	Native Development Kit on työkalu, jota käytetään hyödyksi Android-sovellusten virheentarkistuksessa ja ajamisessa.
HTML	Hypertext Markup Language on merkintäkieli, jonka avulla kuvataan verkkosivujen sisällön rakenne.
.NET	.NET Framework on Microsoftin ohjelmistokomponenttikirjasto, jota Visual Studiolla kehityt ohjelmistot käyttävät.
Silverlight	Microsoft Silverlight on kehitystyökalu Web- ja mobiilisovellusten kehittämiseen. Silverlight on ilmaiseksi ladattava lisäosa.
WPF	Windows Presentation Foundation on kirjasto, joka muodostaa Windows Vistan ja myöhempien versioiden graafisen rajapinnan.
C#	Microsoftin .NET-konseptia varten kehittämä ohjelmointikieli. Lähdekooditiedostot tallennetaan .cs-päätteellä.
iOS	Applen kehittämä käyttöjärjestelmä, jota käytetään vain Applen laitteissa.
Android	Mobiililaitteille suunniteltu käyttöjärjestelmä.
NuGet	Microsoftin kehitysympäristöön tehty pakettienhallintaohjelma.

1 JOHDANTO

Tarjolla on monia erilaisia tekniikoita ja ohjelmointiympäristöjä sovellusten käyttöliittymien kehittämiseen. Käyttöliittymä on tärkeä osa sovellusta ja vaikuttaa sovelluksen käyttökokemukseen merkittävästi.

Opinnäytetyön aiheena on XAML, käyttöliittymien suunnitteluun ja kehittämiseen tarkoitettu merkintäkieli. Opinnäytteekseni teen yksinkertaisen, XAMLin perusasioita käsittelevän esimerkkisovelluksen. Kehitän Xamarinia käyttäen esimerkkisovelluksen Androidille havainnollistamaan XAMLia eri tavoin. Tämä esimerkkisovellus tulee olemaan englannin kielellä, koska se kehitetään opetusmateriaaliksi Mobile Programming -moduuliin ja moduulin opetuskielenä on englanti. Aiheessa minua kiinnostaa eniten mobiiliohjelmointi ja se, missä kaikessa XAMLia käytetään tai voidaan käyttää. Valitsin XAMLin opinnäytetyöni aiheekseni, koska ehdin ICT-projektissa tutustua siihen jo jonkin verran, käytimme sitä paljon projektissamme ja tykkäsin XAMLin tyylistä. Keskityn opinnäytetyössäni mobiiliohjelmointiin ja XAMLiin siltä osin.

Opinnäytetyön tavoitteena on saada tuotettua selkeä ja visuaalinen XAML-esimerkkisovellus mobiiliohjelmoinnin aloittelijoille. Samalla oppia lisää aiheesta ja käyttää XAMLia sujuvasti ja tehokkaasti. Tavoitteeseen päästään käyttämällä useita eri lähteitä ja testailemalla paljon erilaisia elementtejä XAMLilla.

ICT-projekti DigiSport -sovellusta tehdessä opin paljon Xamarinilla mobiiliohjelmoinnista ja olen oppinut projekteissa hyödyntämään erityyppisiä lähteitä ja etsimään tietoa laajemmin kuin ennen. Uskon että tämän työn tekemisen ohessa opin paljon mobiiliohjelmoinnista ja käyttöliittymien kehittämisestä.

Työssäni kerron Xamarinista, Visual studion uudesta 2017-versiosta, XAMLista, mobiiliohjelmoinnissa, C# ja XAMLin eroavaisuuksista ja XAMLin ja HTML eroavaisuuksista. Lisäksi raportoin esimerkkisovelluksen tulokset ja kerron sovelluksen sisällöstä ja ominaisuuksista.

2 XAML

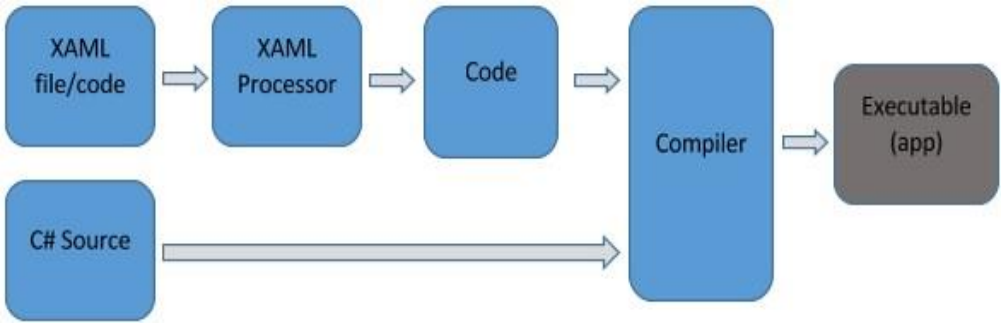
2.1 XAML yleisesti

Kokonaisuudessaan Extensible Application Markup Language eli XAML on Microsoftin kehittämä XML-pohjainen ohjelmointikieli. XAML-tiedostot ovat XML-tiedostoja, joilla on .xaml-pääte. XAMLia käytetään hyödyksi sovelluksien käyttöliittymien koodauksessa ja suunnittelussa. XAMLia käytetään melkein aina liitettynä muiden erilaisten koodikielien kanssa kuten C#, VB.NET ja C++ (Microsoft, 2016a). XAML helpottaa ja yksinkertaistaa sovelluksen käyttöliittymän koodausta merkittävästi, koska koodi saman asian tekemiseksi on huomattavasti lyhyempää ja yksinkertaisempaa kuin esimerkiksi C#-koodi.

XAML mahdollistaa nopean tavan kehittää erilaisia käyttöliittymän elementtejä, kuten esimerkiksi painikkeita, tekstielementtejä, karttoja, kuvia tai animaatioita riippuen käytössä olevista kirjastoista. XAMLilla voidaan hallita sovelluksen ulkoasua ja kehittäjä voi itse suunnitella ja luoda omia elementtejä joita voi käyttää myöhemmin. Lisäksi XAML sisältää paljon muita hyödyllisiä ominaisuuksia kuten tyylejä ja data-malleja. (Microsoft, 2017a).

XAMLia voidaan käyttää eri alustoilla, kuten WPF (Windows Presentation Foundation), Silverlight, Mobiiliohjelmoinnissa ja Windows Store -sovelluksissa. Sitä voidaan käyttää eri .Net framework- ja CLR (common language runtime) -versioissa. (XAML, 2017).

XAMLin prosessin toiminnasta vaiheet alla olevassa kuvassa (Kuva 1): Ensin alustakohtainen XAML-prosessori tulkitsee XAML-tiedoston. Seuraavaksi prosessori muuntaa XAML-elementit sisäiseksi koodiksi. Sisäinen koodi ja C#-koodi ovat sidoksissa toisiinsa luokkamääritelmien perusteella, ja lopuksi .NET-kääntäjä rakentaa sovelluksen ja se on valmis ajettavaksi.



Kuva 1. Kaavio XAMLin prosessista (XAML, 2017).

2.2 XAML-syntaksi

XAMLin syntaksi näyttää melkein samalta kuin HTMLn syntaksi. XAML-syntaksin tärkein asia ovat elementin aloitus- ja lopetusmerkit. Virheet syntaksissa havai-

taan yleensä punaisesta alleviivauksesta. Objektin elementin syntaksi alkaa merkillä < ja loppuu merkkeihin />. Erikoismerkit kuten pienempi kuin, suurempi kuin, et- ja lainausmerkit XAMLissa tulee merkitä kuvan osoittamalla tavalla (Kuva 2):

```
&lt;   <!-- Less than symbol -->
&gt;   <!-- Greater than symbol -->
&amp;  <!-- Ampersand symbol -->
&quot; <!-- Double quote symbol -->
```

Kuva 2. Kuvakaappaus esimerkistä (Microsoft, 2015).

Ominaisuudet, kuten painikkeen teksti määritetään seuraavalla tavalla:

```
<Button Text="Painike"/>
```

Jos elementin sisälle tulee enemmän sisältöä, voidaan myös käyttää seuraavaa tapaa:

```
<Label>Esimerkkitekstiä</Label>
```

Alapuolella on esimerkki yksinkertaisesta XAML-koodista havainnollistamaan XAMLin syntaksia:

```
<ContentPage.Content>
  <StackLayout>
     <Button Text="Esimerkkikoodi"/>
     <Label Text="Syntaksista"/>
   </StackLayout>
</ContentPage.Content>
```

2.3 XAMLin historia

Ennen XAMLia käyttöliittymät kehitettiin pääasiassa käyttäen WinForms-tekniikkaa. Varhaisessa vaiheessa oli keskustelua miltä merkintäkielen kuuluisi näyttää ja millaista sen tulisi olla. Tiedossa oli, että halutaan selkeä ja helppokäyttöinen käyttöliittymämalli, mutta ei tarkkaa tietoa ominaisuuksista joita halutaan lisätä vanhoihin malleihin. Myöhemmän keskustelun pohjalta päädyttiin siihen, että mallin tulisi pohjautua .NET-käytäntöihin ja formaatin tulisi olla pysyvä ja tukea .NET-objekteja. (Anderson, 2003).

XAML kehitettiin tuomaan tukevammat ja paremmin suunnitellut käyttöliittymät sovelluksiin ja yksinkertaistamaan käyttöliittymien yleistä kontrollia kehittäjille. Ensimmäinen julkaisu XAMLista tuli kesäkuussa 2008 Microsoftilta. (Microsoft, 2006).

2.4 XAML ja C#

Sovelluksen koodaus pelkkää C#-kieltä käyttäen on mahdollista, mutta mitä suurempi sovellus ja enemmän koodia, sitä työläämmäksi käyttöliittymän suunnittelu

ja kehitys ilman XAMLia tulee. XAML ja C# tukevat toisiaan, XAML-tiedostoon voidaan koodata käyttöliittymän elementit ja C#-koodissa voidaan luoda ja hallita käyttöliittymäelementtien toimintoja. Jos luodaan XAML-tiedosto nimeltä Painikkeita.xaml, automaattisesti luodaan myös tiedosto Painikkeita.xaml.cs jos koodikieli on C#. Tähän .cs -tiedostoon voidaan kirjoittaa C#-koodia ja hallita XAML-koodissa luotuja elementtejä. Tällöin puhutaan ”code-behind”-tiedostosta. (Microsoft, 2016b). Esimerkkejä tulkiten ja vertaillen XAML-koodi on lyhyempää ja yksinkertaisempaa kuin C#-koodi.

Sovelluksen koodaus pelkkää C#-kieltä käyttäen on mahdollista, mutta mitä suurempi sovellus ja enemmän koodia, sitä työläämmäksi käyttöliittymän suunnittelu ja kehitys ilman XAMLia tulee. Lisäksi jos aletaan käsitellä isompia elementtejä ja muokkailemaan elementtien kokoa tai muuta sellaista, XAML käy aina vain kätevämmäksi.

Painikkeen luonti XAML-koodissa:

```
<Button TextColor="Black" Text="Start" Clicked="ButtonPagelle" BackgroundColor="Blue"/>
```

Painikkeen luonti vaihtoehtoisesti C#-koodissa:

```
Button button = new Button
{
    Text = "Start",
    TextColor = Color.Black,
    BackgroundColor = Color.Blue

};
  button.Clicked += ButtonPagelle;
```

Painikkeen toimintametodi vie sovelluksessa uudelle sivulle (ButtonPage):

```
void ButtonPagelle (object sender, EventArgs e)
{
    this.Navigation.PushAsync(new ButtonPage());
}
```

2.5 Windows Presentation Foundation

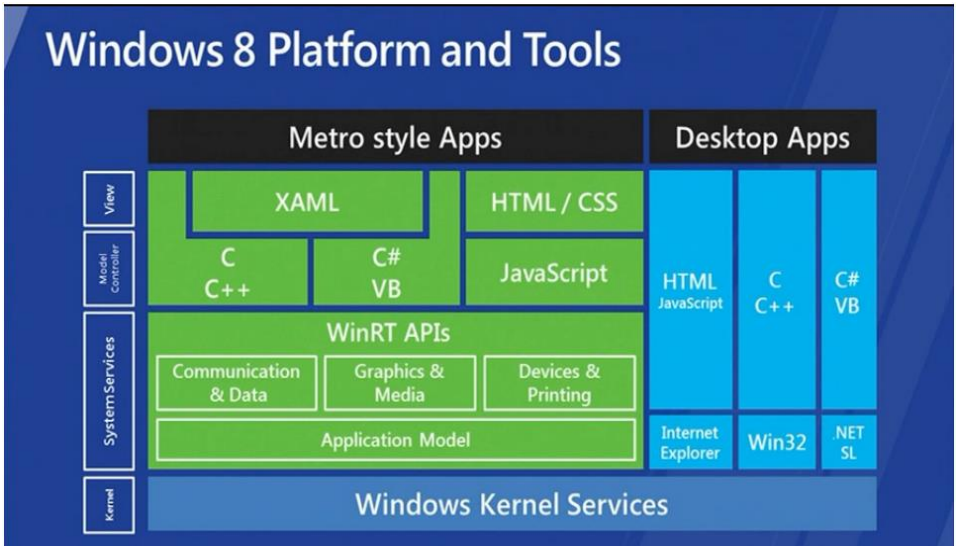
Windows Presentation Foundation (WPF) on graafinen alijärjestelmä, jota käytetään kehittämään ja kääntämään käyttöliittymiä erilaisissa sovelluksissa. Windows Presentation Foundation ja XAML muodostavat hyvän yhdistelmän kehittää ja rakentaa Windows-työpöytäsovelluksia. (Microsoft, 2016d). WPF käyttää .NET-Frameworkia. WPF-sovellukset voidaan rakentaa (build) .exe -muodossa, .dll tai näiden yhdistelmänä. (Microsoft, 2016c).

Rakentamisen jälkeen WPF-sovellus pitää ottaa käyttöön (deploy). Windows ja .NET-Framework sisältävät muutamia käyttöönottotekniikoita, kuten XCopy deployment, Windows Installer deployment ja ClickOnce deployment.

2.6 XAML ja HTML

XAML- ja HTML-merkintäkielet ovat syntaksiltaan lähes samanlaisia, mutta eroavat kuitenkin useissa asioissa ja käyttävät eri teknologioita. XAML käyttää yleensä C#- tai C++-koodikieliä apuna sovelluksen kehittämisessä. HTML taas käyttää JavaScript-teknologiaa. Asiasta on paljon keskustelua ja mielipiteitä kumpi on parempi vaihtoehto ja minkälaisen sovelluksen kehittämiseen. Molemmat ovat joustavia ja soveltuvat moniin asioihin, eroja löytyy riippuen alustasta ja sovelluksen toiminnoista ja käyttötarkoituksesta.

Web-ohjelmointiin HTML soveltuu ehkä paremmin ja XAML soveltuu taas paremmin ulkoasujen hallitsemiseen. Kaikilla Microsoftin alustoilla on ainakin yksi asia yhteistä: XAML. Jotkin alustat tukevat muita käyttöliittymän kehittämiseen tarkoitettuja tekniikoita, mutta XAML on yleisin valinta. (Nixon, 2012). Minun mielestäni tässä on enemmän kyse mielipiteestä, kuin kumpi on oikeasti parempi ja helpompi. Yksi ratkaisevista syistä valita XAML on, että se toimii C#:n kanssa. Alla kuva Windows 8 alustoista ja työkaluista (Kuva 3).



Kuva 3. Havainnollistava kaavio Windowsin työkaluista (Mortensen, 2016).

2.7 Käyttäjäkokemuksia XAMLista

Ensimmäiset ajatukset XAMLia käyttäessäni olivat sekaiset. En osannut syntaksia tai tiennyt oikeastaan mihin kohtaan koodista pitää koskea, että alkaa tapahtua. ICT-projektissa lähdin tekemään Gps-sivua mobiilisovellukseemme. XAML ei ollut minulle vielä tuttu ja C# oli eri tavalla hallussa, lähdin aluksi koodaamaan käyttöliittymää C#-koodiin. Muutaman päivän kuluttua tajusin, että tähän on oltava helpompikin vaihtoehto, ja päätin opiskella XAMLia. Hetken opiskelun jälkeen ymmärsin syntaksin ja perusjutut. Ihastuin XAMLiin ja jatkoin XAMLin opiskelua.

Uusi XAML Previewer helpottaa koodausta entisestään. Previewer antaa mahdollisuuden nähdä koodiin tehdyt muutokset live tilassa. En saanut tätä toimimaan kovan yrityksen jälkeen.

Melkein vuoden ajan XAMLia käyttäneenä ja opiskelleena siitä alkaa saamaan enemmän irti ja se tuntuu todella mukavalta tavalta kehittää ja suunnitella käyttöliittymiä mobiilisovelluksiin.

Alun vaikeuksien jälkeen loppujen lopuksi XAML on omien kokemuksien mukaan helppo oppia. XAML on kätevä, koodia on helppo lukea ja ymmärtää ja internetistä löytyy apua, mikäli ongelmia ilmenee. XAMLissa yllätti sen laajuus ja se, mitä kaikkea sillä on mahdollista tehdä.

3 MENETELMIÄ JA TYÖKALUJA

3.1 Microsoft Visual Studio

Microsoft Visual Studio on Microsoftin ohjelmointiympäristö, jolla voidaan kehittää sovelluksia eri alustoille kuten Windows ja Android. Visual Studiossa voidaan käyttää useita ohjelmointikieliä kuten C# ja C++. Visual Studio sopii opiskelijoille, harrastelijoille ja ammattilaisille. Visual Studio sisältää paljon ominaisuuksia sovellusten kehittämiseen ja virheentarkistukseen. (Microsoft, 2017b).

Visual Studiosta on paljon eri versioita, ensimmäinen julkaistiin 1997 (Microsoft, 2017c). Tässä opinnäytetyössä käytettiin uusinta, mutta vielä Preview-vaiheessa olevaa Visual Studio 2017-versiota. Visual Studio 2017:n aloitusruudusta saa valita suoraan mitkä lisäosat halutaan mukaan. Näitä lisäosia voi ladata ja päivittää vielä myöhemminkin halutessaan samasta aloitusruudusta.

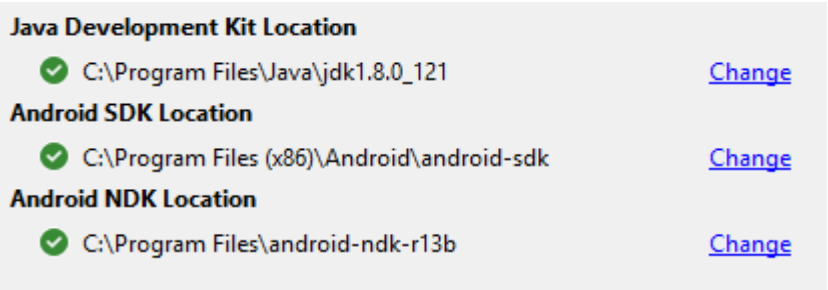
3.2 Xamarin

Xamarin on Microsoftin omistama ohjelmistoyritys ja ohjelmointiympäristö. Xamarinilla voidaan kehittää sovelluksia monille eri mobiilialustoille kuten Android, iOs ja Windows-phone. Xamarinin käyttämä koodikieli on C#. Xamarin on tunnettu cross-platform-teknologiastaan, joka antaa mahdollisuuden kehittää sovelluksia monille eri mobiilialustoille samaa C#-koodia käyttäen. (Jensen, 2014). Tässä opinnäytetyössä mobiilisovelluksen kehittämiseen on käytetty Xamarinia ja sen työkaluja kuten Xamarin.Forms.

Xamarin hyödyntää erilaisia ohjelmointirajapintoja ja NuGet paketteja sovellusten kehityksessä. NuGet on pakettienhallintaohjelma, joka antaa mahdollisuuden ladata paketteja sovellukseen ja huolehtii pakettien päivityksistä. Tässä opinnäytetyössä on käytetty Xamarin.Forms-paketteja ja sen alipaketteja.

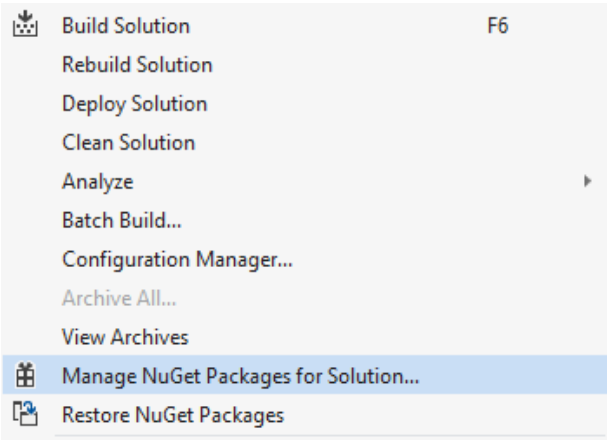
Xamarinilla on suuri yhteisö ongelmatilanteita, keskustelua, blogeja ja tapahtumia varten. Foorumeilla ongelmatilanteisiin voivat vastaavat Xamarinin käyttäjät, sekä ammattilaiset ja antaa neuvoja ongelman ratkaisemiseen.

Xamarinilla työskennellessä on hyvä huolehtia, että NuGet paketit, JDK, SDK ja NDK ovat ajan tasalla ja kuvan osoittamalla tavalla polut oikein (Kuva 8). Näitä pääsee muuttamaan Visual Studiossa Tools -> Options -> Xamarin -> Android Settings. Myös tärkeää on pitää Xamarinin omat päivitykset ajan tasalla.



Kuva 8. JDK, SDK ja NDK tiedostopolut Visual Studiossa, kuvakaappaus.

NuGet paketteja Visual Studiossa pääsee hallitsemaan klikkaamalla projektia ja sieltä Manage NuGet Packages for Solution (Kuva 9).



Kuva 9. Solution-valikko Visual Studiossa, kuvakaappaus.

3.2.1 Xamarin.Forms

Xamarin.Forms on järjestelmäriippumaton käyttöliittymäkirjasto (cross-platform UI toolkit). Xamarin.Forms mahdollistaa ohjelmistokehittäjille helpon tavan luoda käyttöliittymien ulkoasuja, joita voidaan käyttää Androidilla, iOSilla, ja Windows Phonella. (Xamarin, 2016f).

3.3 XAML Previewer

XAML Previewer on uusi Xamarinin työkalu XAML-käyttöliittymäkoodauksen helpottamiseen. XAML Previewer mahdollistaa sovelluksen esikatselun ennen sen rakentamista ja ajamista. Vaatimukset Previewerin käyttämiseen ovat viimeisin Xamarin.Forms NuGet paketti ja Android sovelluksilla JDK versio 1.8 x64. XAML Previewerillä näkee XAML-koodiin tehdyt muutokset heti erillisessä ikkunassa. Previewerillä on mahdollisuus kääntää sovellus puhelimen tai tabletin ruudun kokoiseksi ja myös näyttää sovellus Android tai iOS versiona.

Jos Preview ei toimi, ongelmina saattavat olla seuraavat tekijät: Projekti pitää rakentaa (build) ennen esikatselua, NuGet packaget ja JDK-versio.

3.4 **Testaus**

Käytin mobiilisovelluksen testaukseen ja virheentarkistukseen Visual Studion Android 6.0 – API 23 -emulaattoria ja Samsung Galaxy S7 -puhelinta. Emulaattorin asennus tapahtuu Visual Studion työkaluista Android Emulator Managerilla. Emulaattorin käyttöjärjestelmä ja toiminta ovat lähellä oikean puhelimen käyttöjärjestelmää, joten sovellusta voidaan testata ilman oikeaa laitetta. Puolessa välissä projektia kumminkin huomasin, että emulaattorilla virheentarkistus ja testaus ovat huomattavasti hitaampaa kuin oikealla laitteella. Ongelmat emulaattorin kanssa johtuivat tietokoneen huonosta suorituskyvystä. Tämän jälkeen loppuprojektin ajan käytin testaamiseen omaa puhelintani Samsung Galaxy S7, jossa on Android-käyttöjärjestelmä.

4 XAMARIN.FORMS LAYOUTIT

4.1 Layoutit yleisesti

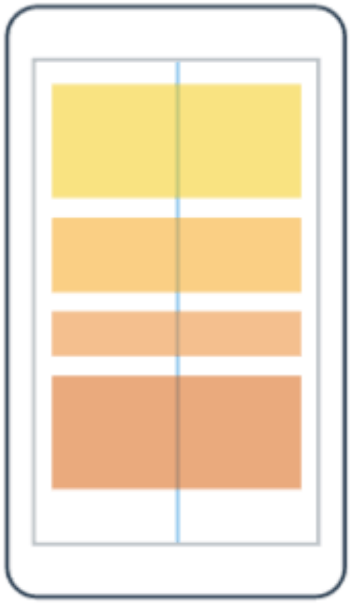
Layoutit eli ulkoasut ovat tärkeä osa sovelluksen käyttöliittymää ja niitä käytetään sovelluksen sisällön hallitsemiseen. Xamarin.Forms sisältää useita layouteja, kuten StackLayout, AbsoluteLayout, RelativeLayout ja Grid (Xamarin, 2016a). Kaikkia layouteja käytetään vähän eri tavalla ja erilaisiin tarkoituksiin, riippuen sovelluksen sisällöstä ja sisällön määrästä. Opinnäytetyön päätarkoituksen ja aiheen mukaisesti kaikki koodiesimerkit tässä luvussa ovat XAMLia. Xamarin.Forms-layouteja voidaan myös käsitellä ja hallita C#-koodissa.

4.2 StackLayout

StackLayoutia käytetään järjestämään sisältöä linjanmukaisesti vaaka- tai pystysuunnassa. Sisällön sijainti ja koko layoutin sisällä määritellään käyttämällä seuraavia ominaisuuksia: HeightRequest, WidhtRequest, HorizontalOptions ja VerticalOptions. StackLayoutia käytetään usein peruslayouttina, järjestämään muita layouteja ruudulla. (Xamarin, 2015a). Alla on koodiesimerkki (Kuva 4) ja havainnollistava kuva StackLayoutin toimintaperiaatteesta (Kuva 5).

```
<StackLayout Orientation="Horizontal">
<Label HorizontalOptions="StartAndExpand" Text="Label"/>
<Button HorizontalOptions="End" Text="Button"/>
</StackLayout>
```

Kuva 4. Koodiesimerkki StackLayoutista (Xamarin, 2015a).



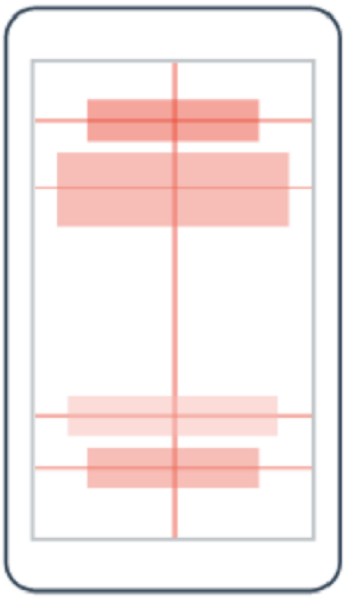
Kuva 5. StackLayoutin toimintaperiaate, 4 elementtiä (Xamarin, 2015a).

4.3 AbsoluteLayout

AbsoluteLayoutia käytetään järjestämään sovelluksen sisältöä tarkasti koordinaattien mukaan haluttuun paikkaan ruudulla. Xamarin.Forms tukee kahta AbsoluteLayoutiin liitettyä ominaisuutta: AbsoluteLayout.LayoutBounds ja AbsoluteLayout.LayoutFlags. LayoutBounds on pilkuilla erotettu neljän arvon lista, joka määrittää rajatun suorakaiteen sijaintia ja ulottuvuuksia. Kaksi ensimmäistä arvoa listassa pitää määrittää numeroina. Kaksi jälkimmäistä arvoa voi merkitä numeroina tai string-muodossa ”AutoSize”. LayoutFlags-ominaisuus määrittää kuinka arvot listassa tulkitaan luotaessa rajattua suorakaidetta. LayoutFlags arvojen nimet ovat All, None, HeightProportional, WidthProportional, SizeProportional, XProportional, Yproportional tai PositionProportional. Mitä tahansa näistä voi käyttää yhdessä pilkuilla erotetulla listalla. (Xamarin, 2015b). Alla on koodiesimerkki (Kuva 6) ja havainnollistava kuva AbsoluteLayoutin toimintaperiaatteesta (Kuva 7).

```
<AbsoluteLayout VerticalOptions="FillAndExpand" Horizonta
 lOptions="FillAndExpand">
<BoxView AbsoluteLayout.LayoutBounds="0.25, 0.25, 0.5, 0.5"
 Color="Blue" AbsoluteLayout.LayoutFlags="All"/>
</AbsoluteLayout>
```

Kuva 6. Koodiesimerkki AbsoluteLayoutista (Xamarin, 2015b).



Kuva 7. AbsoluteLayoutin toimintaperiaate, 4 elementtiä (Xamarin, 2015b).

4.4 RelativeLayout

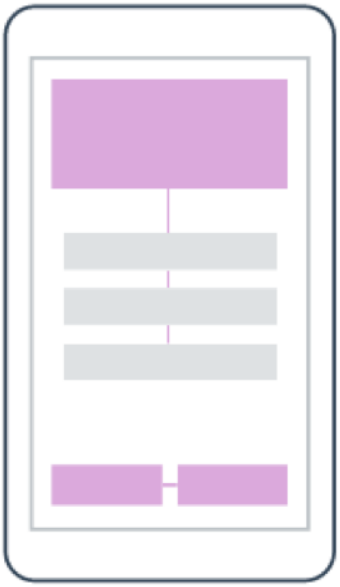
RelativeLayoutia käytetään hallitsemaan layoutin näkymien ja näkymien sukulaisominaisuuksien sijaintia ja kokoa. RelativeLayoutilla ei ole mahdollisuutta sijoittaa elementtejä esimerkiksi pohjalle tai oikealle layoutin reunoille. RelativeLayout tu-

kee elementtien sijoittamista omien rajojensa ulkopuolelle. RelativeLayoutia voidaan käyttää sijoittamaan näkymiä ruudulla suhteellisesti kokonaisvaltaiseen layoutiin tai kahteen eri näkymään.

Näkymien sijainnin ja näkymien sukulaisten hallitsemiseen RelativeLayoutilla käytetään ominaisuutta ConstraintExpressions. ConstraintExpression voi sisältää useita määrityksiä kuten Type, Property, Factor, Constant ja ElementName. Näillä määritellään näkymän koko, sijainti, yhteydet ja sukulaisuudet muiden layoutin näkymien kanssa. RelativeLayoytin näkymien koon määrittämiseen löytyy kaksi vaihtoehtoa, HeightRequest & WidthRequest tai RelativeLayout.WidthConstraint & RelativeLayout.HeightConstraint. Alla on koodiesimerkki (Kuva 8) ja havainnollistava kuva RelativeLayoutin toimintaperiaatteesta (Kuva 9).

```
<RelativeLayout>
<BoxView Color="Red" x:Name="redBox"
 RelativeLayout.YConstraint="{ConstraintExpression
 Type=RelativeToParent,Property=Height,Factor=.15,Con-
 stant=0}"
 RelativeLayout.WidthConstraint="{ConstraintExpression
 Type=RelativeToParent,Property=Width,Factor=1,Constant=0}"
 RelativeLayout.HeightConstraint="{ConstraintExpression
 Type=RelativeToParent,Property=Height,Factor=.8,Con-
 stant=0}"/>
</RelativeLayout>
```

Kuva 8. Esimerkkikoodi RelativeLayoutista (Xamarin, 2015c).



Kuva 9. RelativeLayoutin toimintaperiaate, 6 elementtiä (Xamarin, 2015c).

4.5 **Grid**

Gridiä käytetään järjestämään näkymiä riveihin ja sarakkeisiin. Riveille ja sarakkeille voidaan asettaa koko käyttämällä ominaisuuksia Proportional, Absolute tai

Auto. Proportional muuttaa sarakkeiden ja rivien koot suhteellisesti jäljellä olevan tilan mukaan ruudulla. Absolute muuttaa sarakkeiden ja rivien koot annettujen korkeus- ja leveysarvojen mukaisesti. Auto muuttaa sarakkeiden ja rivien koon mahduttaen sisällön niihin automaattisesti. (Xamarin, 2015d).

Gridillä on kaksi ominaisuutta määrittämään tilan kokoa sarakkeiden ja rivien välissä. ColumnSpacing määrittää tilan sarakkeiden välissä ja RowSpacing määrittää tilan rivien välissä. Alla on koodiesimerkki Gridistä kahdella sarakkeella (Kuva 10).

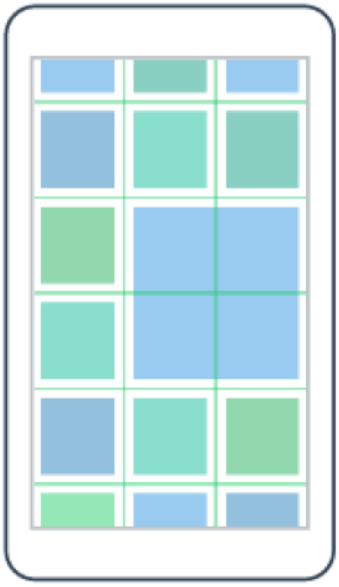
```
<Grid ColumnSpacing="5">
<Grid.ColumnDefinitions>
<ColumnDefinitions Width="*" />
<ColumnDefinitions Width="*" />
</Grid.ColumnDefinitions>
</Grid>
```

Kuva 10. Koodiesimerkki Gridin toimintaperiaatteesta (Xamarin, 2015d).

Usein Gridiä käytettäessä löytyy elementti joka vie enemmän kuin yhden rivin tai sarakkeen alueen, esimerkiksi laskimen koodauksessa. Tähän käytetään Span-ominaisuutta. Alla on koodiesimerkki, jossa painike vie kahden sarakkeen tilan Gridissä (Kuva 11) ja havainnollistava kuva Gridin toimintaperiaatteesta (Kuva 12).

```
<Button Text= "0" Grid.Row="4" Grid.Column="0" Grid.Col-
umnSpan="2" />
```

Kuva 11. Esimerkkikoodi Gridin painikkeista (Xamarin, 2015d).



Kuva 12. Gridin toimintaperiaate (Xamarin, 2015d).

5 XAML-ESIMERKKISOVELLUS

Opinnäytetyön aiheena oli kehittää mahdollisimman visuaalinen XAMLin perusasioita käsittelevä esimerkkisovellus. Kehitin tämän esimerkkisovelluksen Androidille käyttäen lähes pelkästään XAMLia, ainoastaan siirtymäpainikkeiden metodit ovat C#-koodissa. Sovellus sisältää pääsivun, painikesivun, tekstisivun, kaksi layout-sivua ja margin-sivun. Jokaisella sivulla on esillä erilaisia elementtejä ja asioita XAMLiin liittyen. Kaikilla sivuilla on käytetty StackLayoutia. Sivujen vaihto sovelluksessa tehdään hyödyntäen Xamarin.Forms.NavigationPage -luokkaa tähän tapaan:

```
        void ButtonPagelle(object sender, EventArgs e)
        {
            this.Navigation.PushAsync(new ButtonPage());
}
```

Tarkoituksena oli luoda tiivis paketti, josta olisi helppo aloittaa ja ymmärtää XAMLin toimintaperiaatteet ja peruselementit. Sivut ovat enemmän tai vähemmän visuaalisia, mutta varmasti aloittelevalle XAMLin käyttäjälle tästä on ainakin jotain hyötyä. Seuraavissa luvuissa kerron sovelluksen sisällöstä ja esittelen sovelluksessa käytettyjä elementtejä, tekniikoita ja jokaisesta esimerkkisovelluksen sivusta kuvakaappaukset.

Käytin sovelluksessa ainoastaan Xamarin.Forms -pakettia ja sen mukana tulleita alipaketteja. Xamarin.Forms ja siihen kuuluvat paketit ovat pakollisia mobiilisovellusta kehittäessä Xamarinilla. NuGet Package Manager auttaa pakettien hallinnassa ja Xamarinilla on monia erilaisia paketteja eri tarkoituksiin ja erilaisiin sovelluksiin. Esimerkiksi on mahdollista ladata projektiin erilaisia lisäosia, kuten: JavaScript-lisäosia ja Google Maps-lisäosia. Kaikki ladatut paketit ovat listattuna Package.config -tiedostossa ja NuGet Package Managerissa. Alapuolella on esimerkkisovelluksen Package.config -tiedoston sisältö (Kuva 13):

```
<packages>
  <package id="Xamarin.Android.Support.Animated.Vector.Drawable" version="23.3.0" targetFramework="monoandroid70" /
  <package id="Xamarin.Android.Support.Design" version="23.3.0" targetFramework="monoandroid70" />
  <package id="Xamarin.Android.Support.v4" version="23.3.0" targetFramework="monoandroid70" />
  <package id="Xamarin.Android.Support.v7.AppCompat" version="23.3.0" targetFramework="monoandroid70" />
  <package id="Xamarin.Android.Support.v7.CardView" version="23.3.0" targetFramework="monoandroid70" />
  <package id="Xamarin.Android.Support.v7.MediaRouter" version="23.3.0" targetFramework="monoandroid70" />
  <package id="Xamarin.Android.Support.v7.RecyclerView" version="23.3.0" targetFramework="monoandroid70" />
  <package id="Xamarin.Android.Support.Vector.Drawable" version="23.3.0" targetFramework="monoandroid70" />
  <package id="Xamarin.Forms" version="2.3.3.193" targetFramework="monoandroid70" />
</packages>
```

Kuva 13. Esimerkkisovelluksen Package.config -tiedosto, kuvakaappaus.

5.1 Työvaiheet

Aloitin opinnäytetyön tekemisen asentamalla tarvittavia ohjelmistoja omalle tietokoneelleni. Asensin uusimman Visual Studion ja siihen tarvittavat lisäosat kuten

Xamarin ja tarvittavat NuGet paketit tyhjään projektiin. Seuraavaksi asensin uusimmat JDK-, SDK- ja NDK-versiot Javan nettisivuilta. Viimeisenä asensin Android 6.0 – API 23 -emulaattorin ja monien ongelmien jälkeen sain tyhjän projektin ajettua sillä.

Kun kaikki tarvittavat ohjelmistot olivat asennettu, aloin etsiä tietoa lähteistä ja miettiä sovelluksen rakennetta. Koska olin aikaisemminkin käyttänyt StackLayoutia ja tykännyt siitä, päätin käyttää sitä myös tässä sovelluksessa pohjana. Sain melkein Pääsivun valmiiksi XAML Previeweriä apuna käyttäen, kunnes tietokoneen suorituskyky hidastui niin paljon, että en voinut jatkaa projektin tekemistä.

Kovalevyn tyhjennys auttoi ja tietokoneeni oli kuin uusi. Kaikki asennukset uusiksi ja sain kaiken oleellisen toimimaan. Ainoa ominaisuus mitä en saanut enää toimimaan oli XAML Previewer. Sain projektin tehtyä loppuun ilman suurempia ongelmia.

5.2 **Pääsivu**

Esimerkkisovelluksen pääsivulla on kolme tekstielementtiä, kuva XAMLin logosta ja siirtymäpainike, joka vie painikesivulle. Pääsivun sisältöä hallitaan käyttämällä StackLayouteja. Koko sivun kattavan StackLayoutin Margin -arvot ovat 10,15,10,15, joiden takia ruudun vasemmalle ja oikealle reunalle jää 10 pikseliä tyhjää tilaa ja ylä- ja alareunoille jää 15 pikseliä tyhjää tilaa. Margin eli marginaali selkeyttää sivun sisältöä ja se näyttää siistimmältä, kun sisältö ei ole kiinni ruudun reunoissa.

Ensimmäinen tekstielementti pääsivulla on otsikko, jolle on määritelty teksti, fontin koko ja tekstin väri. Pääsivulla on käytetty ominaisuuksia Text, VerticalOptions, HorizontalOptions, FontSize ja TextColor.

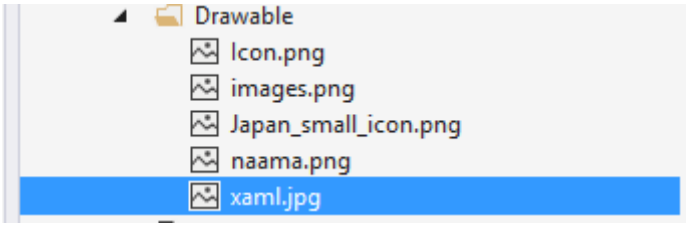
Text määrittää elementtien tekstisisällön. VerticalOptions määrittää elementin sijainnin layoutin sisällä pystysuuntaisesti. HorizontalOptions määrittää elementin sijainnin layoutin sisällä vaakasuuntaisesti. Näiden arvoiksi voidaan antaa Start, Center tai End (Xamarin, 2016b). Pääsivun otsikko on määritetty pystysuunnassa alkuun ja vaakasuunnassa keskelle.

FontSize määrittää elementin tekstin koon ja arvo voidaan antaa muodossa Large, Small, Medium tai kokonaislukuna. Textcolor määrittää tekstin värin. Perusvärit kuten valkoinen voidaan määrittää antamalla arvoksi esimerkiksi White. Valkoisen värin tekstille voi määrittää myös värikoodilla #FFFFFF. Värikoodeilla voidaan antaa elementeille tarkempia väriarvoja, jos halutaan hienompia värejä, esimerkiksi hopea (Computer Hope, 2017).

Otsikon jälkeinen sisältö on eroteltu kahden eri StackLayout -elementin sisään. Ensimmäisen layoutin sisällä on yksi tekstielementti, jonka fontin kooksi on määritetty arvo Medium ja tekstin väriksi White. Molempien StackLayoutien Padding -arvoiksi on määritetty 20,15,20,15. Toisen StackLayoutin sisältö on sijoitettu pys-

tysuuntaisesti loppuun ja laajentumaan niin paljon kuin layoutin rajat antavat. Toisen layoutin sisällä on siirtymäpainike ja painikkeen yläpuolella otsikkona tekstielementti. Tekstielementin fontin kooksi on määritetty Medium ja väriksi White. Painikkeen tekstin väriksi on määritetty Black ja taustaväri määritetty värikoodilla #FF008FF, joka vastaa väriä AliceBlue. Painiketta painaessa kutsutaan pääsivun C#-koodissa olevaa ButtonPagelle-metodia, joka siirtää sovelluksen painikesivulle.

Kahden pienemmän StackLayoutin välissä ja keskellä sivua on XAMLin logo, kuvaelementti. Kuvaelementti lisätään sivulle käyttämällä Image-luokkaa ja määrittämällä kuvan nimi arvoksi Source-attribuutille esimerkiksi xaml.jpg. Jotta kuvaa voisi käyttää projektissa, se on lisättävä projektin Drawable-kansioon (Kuva 14).



Kuva 14. Projektin Drawable kansion sisältö.

Pääsivun näkymästä on kuvakaappaus alapuolella (Kuva 15) ja pääsivun XAML-koodista on kuvakaappaus liitteissä (Liite 1).

Kuva 15. Esimerkkisovelluksen pääsivun näkymä, kuvakaappaus.

5.3 Painikesivu

Esimerkkisovelluksen painikesivulla käsitellään painikkeiden erilaisia ominaisuuksia. Painikesivulla on yhteensä neljä painiketta. Kolme eri tavalla tehtyä painiketta ja siirtymäpainike. Button eli painike on yksi Xamarin.Forms-luokista. Button-luokka sisältää monia ominaisuuksia kuten FontSize, Text, TextColor, Image ja Clicked. Näillä ominaisuuksilla voidaan muuttaa painikkeen ulkonäköä ja tapahtumia. Clicked-ominaisuus määrittää painikkeen tapahtumat ja kutsuu määritettyä metodia painiketta painaessa. (Xamarin, 2016e). Esimerkiksi painikesivun alimmainen painike kutsuu TekstiPagelle-metodia, joka vie seuraavalle sivulle (Kuva 16).

```
void TekstiPagelle(object sender, EventArgs e)
{
    this.Navigation.PushAsync(new TextPage());
}
```

Kuva 16. Painikesivun siirtymäpainikkeen metodi, kuvakaappaus.

Tapahtuma määritetään painike-elementin sisään tähän tyyliin:

<Button Clicked="KutsuuMetodia"/>

Painikesivun ensimmäinen painike on tavallinen painike, ilman mitään muokkauksia. Painikkeen oletusväri on harmaa ja painikkeen tekstin oletusväri on valkoinen. Sen yläpuolella tekstielementissä XAML-koodi, jolla painikkeen luonti tapahtuu.

Toisen painikkeen kokoa on muutettu käyttäen HeightRequest- ja WidthRequest-ominaisuuksia. HeightRequest määrittää painikkeen korkeuden ja WidthRequest sen leveyden, arvot voidaan antaa kokonaislukuina. Tämän painikkeen korkeus on 70 pikseliä ja leveys 200 pikseliä. Annetuista arvoista riippumatta, painike ei suurene layoutin ulkopuolelle.

Kolmannen painikkeen sisään on upotettu kuva käyttämällä image-ominaisuutta. Painike laajentuu automaattisesti kuvan koon mukaisesti, silti pysyen layoutin sisällä. Sen yläpuolella on tekstielementissä XAML-koodi, jolla kuva on saatu painikkeen sisään.

Painikesivun näkymästä on kuvakaappaus alapuolella (Kuva 17) ja pääsivun XAML-koodista on kuvakaappaus liitteissä (Liite 2).

Kuva 17. Painikesivun näkymä, kuvakaappaus.

5.4 Tekstisivu

Tekstisivulla on yhteensä kahdeksan tekstielementtiä ja siirtymäpainike. Valkoiset tekstielementit toimivat ikään kuin otsikkoina eri attribuuteilla muokatuille punaisille tekstielementeille. Tekstisivulla havainnollistetaan Label-luokan käyttöä ja sen muutamia ominaisuuksia.

Tekstisivulla on kuusi StackLayout-elementtiä. Yksi StackLayout kattaa koko sivun ja sen sisältö on määrätty sijoittumaan vaakasuuntaisesti keskelle ruutua. Loput viisi toimivat ryhmittäjinä tekstielementeille ja siirtymäpainikkeelle. Tämä selkeyttää sivua ja on helppo tapa erotella tekstielementit toisistaan.

Jokaisen punaisen tekstielementin fonttikoko on 20 ja jokaisen valkoisen tekstielementin fonttikoko on 15. Ensimmäisessä esimerkissä esitetään, miten tavallinen tekstielementti luodaan. Toisessa esimerkissä esitetään, miten Label-luokan FontSize-ominaisuutta käytetään tekstielementin fonttikoon muuttamiseen. Kolmannessa esimerkissä esitetään, miten tekstielementin tekstin väriä voidaan muuttaa XAMLissa. Neljännessä esimerkissä valkoisen tekstielementin fontin tyyli on Italic, ja punaisen tekstielementin fontin tyyli on Bold. Nämä ovat FontAttributes-ominaisuuden ainoat saatavilla olevat tyylit Xamarin.Forms-fonteissa (Xamarin, 2016d).

Tekstisivun näkymästä on kuvakaappaus alapuolella (Kuva 18) ja tekstisivun XAML-koodista on kuvakaappaus liitteissä (Liite 3).

Kuva 18. Esimerkkisovelluksen tekstisivun näkymä, kuvakaappaus.

5.5 Layout-sivut

Layout-sivuja on kaksi kappaletta. Sivuilla tuodaan esille StackLayoutin toiminta-periaatteita ja se, kuinka sitä voidaan käyttää XAMLissa. Jo sivun pohjaa suunnitellessa kannattaa miettiä, mikä ulkoasuista olisi paras. Vaihtoehtoja on muutamia, kuten StackLayout, RelativeLayout, AbsoluteLayout ja Grid. Sivun sisältöä on helppo jakaa ja hallita ulkoasujen avulla. Hyvä tapa on ensimmäiseksi luoda esimerkiksi koko sivun kattava StackLayout ja laittaa sivun sisältö sen sisälle. Tällä tavoin saadaan kaikelle sivun sisällölle sama marginaali, määräykset ja linjaukset. Se estää myös sisältöä karkaamasta ruudun rajojen ulkopuolelle ja helpottaa käyttöliittymän kehittämistä.

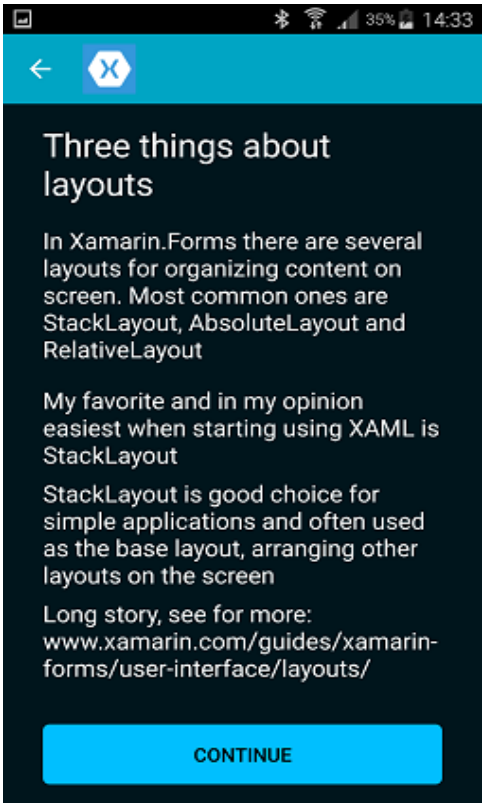
Ensimmäisellä layout-sivulla kerrotaan muutama asia layouteista. Toisella layout-sivulla avataan hieman StackLayoutin toimintaperiaatetta havainnollistavalla kuvalla ja koodiesimerkillä mahdollisesta StackLayout-elementin sisäisestä XAML-koodista.

Ensimmäisellä layout-sivulla on yksi koko sivun kattava StackLayout-elementti ja neljä pienempää sen sisässä olevaa StackLayout-elementtiä. Tekstielementit on jaoteltu kolmen StackLayout-elementin sisään ja siirtymäpainike oman

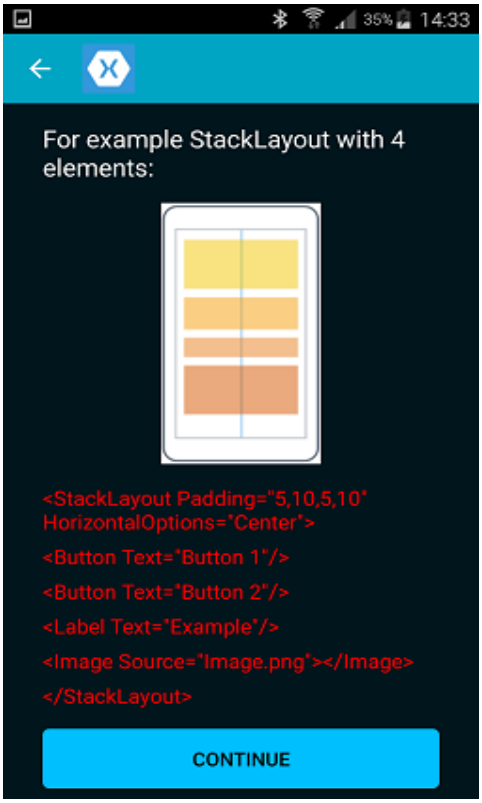
StackLayout-elementin sisään. Sivun sisältö on määrätty sijoittumaan vaakasuuntaisesti keskelle ruutua antamalla HorizontalOptions-ominaisuuden arvoksi Center.

Toisella layout-sivulla on yhteensä viisi StackLayout-elementtiä. Yksi koko sivun kattava ja neljä sen sisällä olevaa StackLayout-elementtiä, jotka hallitsevat sivun sisältöä. Sivun kuvassa on neljä eri kokoista StackLayout-elementtiä ja se esittää miten StackLayout toimii käytännössä.

Alapuolella on molempien esimerkkisovelluksen layout-sivujen näkymät, ensimmäinen sivu (Kuva 19) ja toinen sivu (Kuva 20). Layout-sivujen XAML-koodit ovat liitteissä, ensimmäisen sivu (Liite 4) ja toinen sivu (Liite 5).



Kuva 19. 1. Layout-sivu, kuvakaappaus



Kuva 20. 2. Layout-sivu, kuvakaappaus.

5.6 Margin-sivu

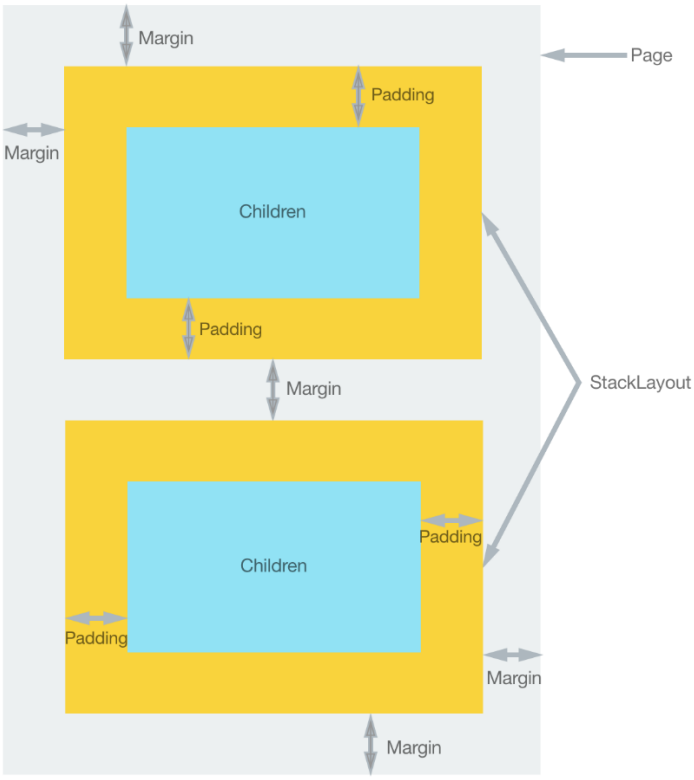
Margin-sivulla selitetään miten margin- ja padding-ominaisuudet toimivat XAMLissa, niiden merkintätavasta, mihin ne vaikuttavat ja mihin suuntaan mikäkin annettu arvo vaikuttaa ruudulla.

Margin määrittää etäisyyden elementin ja sen viereisten elementtien välillä. Marginia käytetään antamaan tilaa elementtien välillä, hallitsemaan elementtien sijaintia ja tekemään tilaa ruudun reunoille marginaalin tavoin.

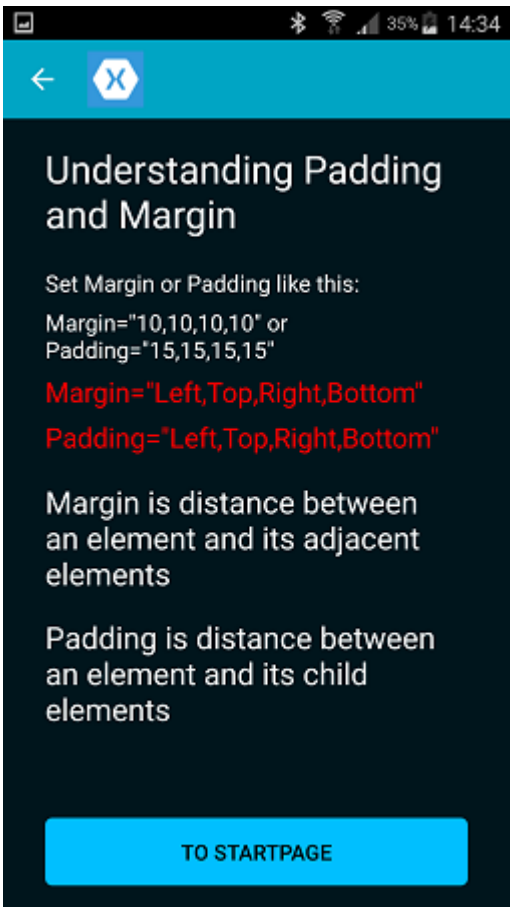
Padding määrittää etäisyyden elementtien ja niiden lapsielementtien välillä. Padding toimii ikään kuin sivun sisällön pehmusteena ja estää elementtejä törmäämästä toisiinsa.

Margin- ja padding-arvoja määritettäessä pitää muistaa, että margin -arvot ovat additiivisia. Esimerkiksi jos kahden vierekkäisen elementin margin -arvoiksi on määritetty 20 pikseliä, elementtien etäisyys toisistaan on 40 pikseliä. Myös margin ja padding ovat additiivisia jos molempia käytetään. Tässä tapauksessa elementin ja sisällön välinen etäisyys on margin plus padding. (Xamarin, 2016c).

Seuraavalla sivulla on esimerkkisovelluksen margin-sivun näkymä (Kuva 21) ja marginsivun XAML-koodi liitteissä (Liite 6). Alla oleva kuva havainnollistaa margin- ja padding-ominaisuuksien toimintatapaa (Kuva 22).



Kuva 21. Margin- ja padding-ominaisuudet.



Kuva 22. Margin-sivun näkymä, kuvakaappaus.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli luoda XAMLin perusasioita havainnollistava mobiilisovellus Androidille ja oppia käyttämään XAMLia sujuvasti. Sovellus ja opinnäytetyö on tarkoitettu opetusmateriaaliksi Mobile Programming-moduuliin tietojenkäsittelyn koulutuksessa. Sovellukseen kuuluu 6 sivua, pääsivu, painikesivu, tekstisivu, kaksi layout-sivua ja margin-sivu. Tämä projekti oli minun ensimmäinen kokemukseni suunnitella ja toteuttaa mobiilisovellus itsenäisesti. Aikaisemmin olen toiminut ryhmässä.

Opinnäytetyö tehtiin kevätlukukaudella 2017. Projektin aikana pidettiin yhteyttä sähköpostin välityksellä tilaajan kanssa ja keskusteltiin projektin etenemisestä ja sisällöstä.

Sovelluksesta olisi voinut tulla laajempi ja hienompi, mutta kaiken kaikkiaan olen melko tyytyväinen työn tulokseen ja mielestäni opin ainakin paljon mobiiliohjelmoinnista, käyttöliittymien suunnittelusta ja kehittämisestä ja tarjolla olevista tekniikoista ja menetelmistä. Sovelluksesta ja oppimateriaalista saa kattavan käsityksen XAMLista, mobiiliohjelmoinnista, tarjolla olevista työkaluista, käyttöliittymistä, käyttöliittymien suunnittelusta ja kehittämisestä. Esimerkkisovellusta voidaan jatkokehittää lisäämällä siihen jonkinlainen valikko ja lisää sivuja. Jos aikaa olisi ollut enemmän, olisin voinut parannella sovelluksen ulkoasua ja tehdä sen sisällöstä kattavamman. Myös tekstiä opinnäytetyön raportissa voisi olla vähän enemmän, mutta kirjoittamista, aiheiden valikointia ja tärkeiden asioiden poimintaa lähteistä vaikeuttivat englanninkieliset lähteet.

Kohtasin opinnäytetyötä tehdessä useita ongelmia. Useimmat ongelmat olivat Xamarinin kanssa. Olen käyttänyt Xamarinia aikaisemminkin ja kohdannut ennenkin monia ongelmia NuGet pakettejen, Xamarinin päivityksien ja emulaattoreiden kanssa. Xamarinin foorumeilta ja muualta internetistä löytyy tietoa näistä ongelmista.

Mielestäni on hyvä, jos opinnäytetyötäni voidaan käyttää opetusmateriaalina ja hyödyntää Mobile Programming-moduulissa.

LÄHTEET

Anderson, C. (2003). A Brief History of XAML. Blogijulkaisu 31.10.2003. Haettu 25.2.2017 osoitteesta http://xml.coverpages.org/ms-xaml.html

Computer Hope. (2017). HTML color codes and names. Haettu 14.3.2017 osoitteesta http://www.computerhope.com/htmcolor.htm

Jensen, D. (2014). Building Android Apps in C# With Xamarin. Blogijulkaisu 26.6.2014. Haettu 23.2.2017 osoitteesta https://code.tutsplus.com/courses/building-android-apps-in-c-with-xamarin/lessons/what-is-xamarin

Microsoft. (2006). XAML Object Mapping Specification. PDF-tiedosto. Haettu 27.2.2017 osoitteesta http://download.microsoft.com/download/0/A/6/0A6F7755-9AF5-448B-907D-13985ACCF53E/%5BMS-XAML%5D.pdf

Microsoft. (2015). How to: Use Special Characters in XAML. Haettu 20.2.1027 osoitteesta https://msdn.microsoft.com/en-us/library/aa970677(v=vs.110).aspx

Microsoft. (2016a). XAML Overview (WPF). Haettu 20.2.2017 osoitteesta https://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx

Microsoft. (2016b). The relationship between XAML and code-behind files. Haettu 20.2.2017 osoitteesta https://msdn.microsoft.com/en-us/library/cc295302.aspx

Microsoft. (2016c). Building a WPF Application (WPF). Haettu 25.2.2017 osoitteesta https://msdn.microsoft.com/en-us/library/aa970678(v=vs.110).aspx

Microsoft. (2016d). Deploying a WPF Application (WPF). Haettu 26.2.2017 osoitteesta
https://msdn.microsoft.com/en-us/library/aa969776(v=vs.110).aspx

Microsoft. (2017a). XAML Overview. Haettu 20.2.2017 osoitteesta
https://msdn.microsoft.com/en-us/library/cc189036(VS.95).aspx

Microsoft. (2017b). Visual Studio Community. Haettu 29.2.2017 osoitteesta
https://www.visualstudio.com/vs/community/

Microsoft. (2017c). Visual Studio 2017 RC. Haettu 27.2.2017 osoitteesta
https://www.visualstudio.com/en-us/news/releasenotes/vs2017-relnotes

Mortensen, P. (2016). Windows 8 Platform and Tools. Foorumijulkaisu 3.3.2016. Haettu 22.2.2017 osoitteesta http://stackoverflow.com/questions/7416826/how-does-windows-8-runtime-winrt-windows-store-apps-windows-10-universal-ap

New York Wrox. (2012). Professional Windows 8 Programming : Application Development with C# and XAML. HAMKin E-kirja.

Nixon, J. (2012). Windows 8: Top 10 Reasons why I choose XAML over HTML5. Blogijulkaisu toukokuu 2012. Haettu 26.2.2017 osoitteesta http://blog.jerrynixon.com/2012/05/windows-8-why-i-choose-xaml-metro-over.html

Tutorials Point (2017). XAML – Overview. Haettu 23.2.2017 osoitteesta https://www.tutorialspoint.com/xaml/xaml_overview.htm

Xamarin. (2015a). StackLayout. Haettu 19.2.2017 osoitteesta https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/stack-layout/

Xamarin. (2015b). AbsoluteLayout. Haettu 20.2.2017 osoitteesta https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/absolute-layout/

Xamarin. (2015c). RelativeLayout. Haettu 20.2.2017 osoitteesta https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/relative-layout/

Xamarin. (2015d). Grid. Haettu 21.2.2017 osoitteesta https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/grid/

Xamarin. (2016a). Layouts. Haettu 19.2.2017 osoitteesta https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/

Xamarin. (2016b). Fonts. Haettu 16.3 osoitteesta https://developer.xamarin.com/guides/xamarin-forms/user-interface/text/fonts/

Xamarin. (2016c). Margin and Padding. Haettu 18.3.2017 osoitteesta https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/margin-and-padding/

Xamarin. (2016d). Label. Haettu 14.3.2017 osoitteesta https://developer.xamarin.com/api/type/Xamarin.Forms.Label/

Xamarin. (2016e). Button. Haettu 14.3.2017 osoitteesta https://developer.xamarin.com/api/type/Xamarin.Forms.Button/

Xamarin. (2016f). Xamarin.Forms. Haettu 6.4.2017 osoitteesta https://developer.xamarin.com/guides/xamarin-forms/

Esimerkkisovellus. (2017) Opinnäytetyön esimerkkisovelluksen koodit osoitteessa https://bitbucket.org/LauriAleksi/virallinen/src

LIITTEET

Pääsivun XAML-koodi

Liite 1.

```
<StackLayout Margin="10,15,10,15">
    <Label Text="Introduction to XAML" VerticalOptions="Start" HorizontalOptions="Center" FontSize="30" TextColor="White"/>
    <StackLayout Padding="20,15,20,15" HorizontalOptions="Center">
        <Label Text="Welcome to XAML basics tutorial. Im going to tell you some basic things about XAML" FontSize="Medium" TextColor="White"/>
    </StackLayout>
    <Image Source="xaml.jpg"></Image>
    <StackLayout Padding="20,15,20,15" HorizontalOptions="Center" VerticalOptions="EndAndExpand">
        <Label Text="Press the button to start!" FontSize="Medium" TextColor="White"/>
        <Button TextColor="Black" Text="Start" Clicked="ButtonPagelle" BackgroundColor="#FF00BFFF"/>
    </StackLayout>
</StackLayout>
```

Painikesivun XAML-koodi

Liite 2.

```
<StackLayout Margin="10,15,10,10">
    <StackLayout Padding="20,10,20,10" VerticalOptions="Center" HorizontalOptions="Center">
        <Label TextColor="White" FontSize="18" Text="&lt;Button Text=&quot;Basic button&quot;/&gt;" />
        <Button Text="Basic button"/>
    </StackLayout>
    <StackLayout Padding="20,7,20,10" VerticalOptions="Center" HorizontalOptions="Center">
        <Label TextColor="White" FontSize="18" Text="You can change size of the button using HeightRequest and WidthRequest" />
        <Button Text="Button size changed" HeightRequest="70" WidthRequest="200"></Button>
    </StackLayout>
    <StackLayout Padding="20,7,20,5" VerticalOptions="Center" HorizontalOptions="Center">
        <Label HorizontalOptions="Center" TextColor="White" FontSize="18" Text="Adding image to a button: Image=&quot;image.png&quot;" />
        <Button Image="naama.png"></Button>
    </StackLayout>
    <StackLayout Padding="20,5,20,5" VerticalOptions="EndAndExpand">
        <Button TextColor="Black" Text="Continue" Clicked="TekstiPagelle" BackgroundColor="#FF00BFFF"/>
    </StackLayout>
</StackLayout>
```

Tekstisivun XAML-koodi

Liite 3.

```
<StackLayout Margin="10,10,10,10" HorizontalOptions="Center">
    <StackLayout Padding="20,15,20,15">
        <Label FontSize="15" TextColor="White" Text="You can do basic label like this:"/>
        <Label FontSize="20" TextColor="Red" Text="&lt;Label Text=&quot;Your Label&quot;/&gt;"/>
    </StackLayout>
    <StackLayout Padding="20,15,20,15">
        <Label FontSize="15" TextColor="White" Text="Change fontsize of label like this:"/>
        <Label FontSize="20" TextColor="Red" Text="FontSize=&quot;20&quot;"/>
    </StackLayout>
    <StackLayout Padding="20,15,20,15">
        <Label FontSize="15" TextColor="White" Text="Change color of label like this:"/>
        <Label FontSize="20" TextColor="Red" Text="TextColor=&quot;Red&quot;"/>
    </StackLayout>
    <StackLayout Padding="20,15,20,15">
        <Label FontSize="15" FontAttributes="Italic" TextColor="White" Text="Change fontattributes like this:"/>
        <Label FontSize="20" FontAttributes="Bold" TextColor="Red" Text="FontAttributes=&quot;Italic&quot;"/>
    </StackLayout>
    <StackLayout Padding="20,5,20,5" VerticalOptions="EndAndExpand">
        <Button TextColor="Black" Text="Continue" Clicked="LayoutPagelle" BackgroundColor="#FF00BFFF"/>
    </StackLayout>
</StackLayout>
```

Painikesivun XAML-koodi

Liite 4.

```
<StackLayout Margin="10,10,10,10">
    <StackLayout Padding="20,5,20,5">
        <Label TextColor="White" FontSize="18" Text="For example StackLayout with 4 elements:"/>
    </StackLayout>
    <StackLayout Padding="20,5,20,5">
        <Image Source="images.png" HeightRequest="210"></Image>
    </StackLayout>
    <StackLayout Padding="20,5,20,5">
        <Label FontSize="15" TextColor="Red" Text="&lt;StackLayout Padding=&quot;5,10,5,10&quot; HorizontalOptions=&quot;Center&quot;&gt;"/>
        <Label FontSize="15" TextColor="Red" Text="&lt;Button Text=&quot;Button 1&quot;/&gt;"/>
        <Label FontSize="15" TextColor="Red" Text="&lt;Button Text=&quot;Button 2&quot;/&gt;"/>
        <Label FontSize="15" TextColor="Red" Text="&lt;Label Text=&quot;Example&quot;/&gt;"/>
        <Label FontSize="15" TextColor="Red" Text="&lt;Image Source=&quot;Image.png&quot;&gt;&lt;/Image&gt;"/>
        <Label FontSize="15" TextColor="Red" Text="&lt;/StackLayout&gt;"/>
    </StackLayout>
    <StackLayout Padding="20,5,20,5" VerticalOptions="EndAndExpand">
        <Button TextColor="Black" Text="Continue" Clicked="MarginPagelle" BackgroundColor="#FF00BFFF"/>
    </StackLayout>
</StackLayout>
```

Marginsivun XAML-koodi

Liite 5.

```
<StackLayout Margin="10,10,10,10" HorizontalOptions="Center">
    <StackLayout Padding="20,7,20,7">
        <Label FontSize="25" TextColor="White" Text="Understanding Padding and Margin"/>
    </StackLayout>
    <StackLayout Padding="20,7,20,7">
        <Label FontSize="15" TextColor="White" Text="Set Margin or Padding like this:"/>
        <Label FontSize="15" TextColor="White" Text="Margin=&quot;10,10,10,10&quot; or Padding=&quot;15,15,15,15&quot;"/>
        <Label FontSize="18" TextColor="Red" Text="Margin=&quot;Left,Top,Right,Bottom&quot;"/>
        <Label FontSize="18" TextColor="Red" Text="Padding=&quot;Left,Top,Right,Bottom&quot;"/>
    </StackLayout>
    <StackLayout Padding="20,5,20,5">
        <Label FontSize="20" TextColor="White" Text="Margin is distance between an element and its adjacent elements"/>
    </StackLayout>
    <StackLayout Padding="20,5,20,5">
        <Label FontSize="20" TextColor="White" Text="Padding is distance between an element and its child elements"/>
    </StackLayout>
    <StackLayout Padding="20,5,20,5" VerticalOptions="EndAndExpand">
        <Button TextColor="Black" Text="To startpage" Clicked="MainPagelle" BackgroundColor="#FF00BFFF"/>
    </StackLayout>
</StackLayout>
```

Pääsivun XAML-koodi

Liite 6.

```
<StackLayout Margin="10,15,10,15">
    <Label Text="Introduction to XAML" VerticalOptions="Start" HorizontalOptions="Center" FontSize="30" TextColor="White"/>
    <StackLayout Padding="20,15,20,15" HorizontalOptions="Center">
        <Label Text="Welcome to XAML basics tutorial. Im going to tell you some basic things about XAML" FontSize="Medium" TextColor="White"/>
    </StackLayout>
    <Image Source="xaml.jpg"></Image>
    <StackLayout Padding="20,15,20,15" HorizontalOptions="Center" VerticalOptions="EndAndExpand">
        <Label Text="Press the button to start!" FontSize="Medium" TextColor="White"/>
        <Button TextColor="Black" Text="Start" Clicked="ButtonPagelle" BackgroundColor="#FF00BFFF"/>
    </StackLayout>
</StackLayout>
```